

What Is Claimed Is:

- 1 1. A method to automate isolation of native code within a computer
2 program that has been compiled to a platform-independent code, the method
3 comprising:
4 receiving a library containing a native code sub-routine, wherein the native
5 code sub-routine provides a service to the computer program;
6 analyzing the library to determine a defined symbol name for the native
7 code sub-routine;
8 creating a proxy sub-routine for the native code sub-routine, wherein the
9 proxy sub-routine forms a link to the native code sub-routine;
10 placing the proxy sub-routine into a new library using the defined symbol
11 name of the native code sub-routine as a symbol name for the proxy sub-routine;
12 running the native code sub-routine in a first process;
13 executing the platform-independent code in a second process; and
14 invoking the native code sub-routine in the first process by calling the
15 proxy sub-routine from the platform-independent code in the second process.
- 1 2. The method of claim 1, further comprising:
2 providing a proxy platform-independent native interface (PINI) to the
3 library containing the native code sub-routine; and
4 transparently transforming local PINI calls into calls to the proxy PINI,
5 wherein transforming local PINI calls into calls to the proxy PINI
6 preserves an original control flow, and
7 wherein upcalls and downcalls are guaranteed to be executed by a same
8 thread of a process that executes the native code sub-routine.

1 3. The method of claim 1, wherein analyzing the library to determine
2 the defined symbol name includes analyzing the library to determine call
3 arguments for the defined symbol name.

1 4. The method of claim 3, wherein analyzing the library to determine
2 call arguments for the defined symbol name is accomplished at runtime by
3 analyzing a current call frame.

1 5. The method of claim 3, further comprising copying call arguments
2 from the proxy sub-routine to a call to the native code sub-routine.

1 6. The method of claim 3, further comprising returning a result value
2 from the native code sub-routine to the proxy sub-routine.

1 7. The method of claim 1, wherein operations in the first process are
2 isolated from memory and other system resources belonging to the second process
3 so that an error in the first process does not, one of, corrupt memory belonging to
4 the second process and interfere with the second process in any way.

1 8. The method of claim 1, wherein the proxy sub-routine and the
2 native code sub-routine communicate through inter-process communication.

1 9. The method of claim 1, wherein forming the link to the native code
2 sub-routine includes translating a data element from a first address width in the
3 computer program to a second address width in the native code sub-routine.

1 12. The computer-readable storage medium of claim 10, wherein
2 analyzing the library to determine the defined symbol name includes analyzing the
3 library to determine call arguments for the defined symbol name.

1 13. The computer-readable storage medium of claim 12, wherein
2 analyzing the library to determine call arguments for the defined symbol name is
3 accomplished at runtime by analyzing a current call frame.

1 14. The computer-readable storage medium of claim 12, the method
2 further comprising copying call arguments from the proxy sub-routine to a call to
3 the native code sub-routine.

1 15. The computer-readable storage medium of claim 12, the method
2 further comprising returning a result value from the native code sub-routine to the
3 proxy sub-routine.

1 16. The computer-readable storage medium of claim 10, wherein
2 operations in the first process are isolated from memory and other system
3 resources belonging to the second process so that an error in the first process does
4 not, one of, corrupt memory belonging to the second process and interfere with
5 the second process in any way.

1 17. The computer-readable storage medium of claim 10, wherein the
2 proxy sub-routine and the native code sub-routine communicate through inter-
3 process communication.

1 18. The computer-readable storage medium of claim 10, wherein
2 forming the link to the native code sub-routine includes translating a data element
3 from a first address width in the computer program to a second address width in
4 the native code sub-routine.

1 19. An apparatus that facilitates automated isolation of native code
2 within a computer program that has been compiled to a platform-independent
3 code, the apparatus comprising:

4 a receiving mechanism that is configured to receive a library containing a
5 native code sub-routine, wherein the native code sub-routine provides a service to
6 the computer program;

7 an analyzing mechanism that is configured to analyze the library to
8 determine a defined symbol name for the native code sub-routine;

9 a creating mechanism that is configured to create a proxy sub-routine for
10 the native code sub-routine, wherein the proxy sub-routine forms a link to the
11 native code sub-routine;

12 a placing mechanism that is configured to place the proxy sub-routine into
13 a new library using the defined symbol name of the native code sub-routine as a
14 symbol name for the proxy sub-routine;

15 a running mechanism that is configured to run the native code sub-routine
16 in a first process;

17 an executing mechanism that is configured to execute the platform-
18 independent code in a second process; and

19 an invoking mechanism that is configured to invoke the native code sub-
20 routine in the first process by calling the proxy sub-routine from the platform-
21 independent code in the second process.

1 20. The apparatus of claim 19, further comprising:
2 a providing mechanism configured to provide a proxy platform-
3 independent native interface (PINI) to the library containing the native code sub-
4 routine; and
5 a transforming mechanism that is configured to transparently transform
6 local PINI calls into calls to the proxy PINI,
7 wherein transforming local PINI calls into calls to the proxy PINI
8 preserves an original control flow, and
9 wherein upcalls and downcalls are guaranteed to be executed by a same
10 thread of a process that executes the native code sub-routine..

1 21. The apparatus of claim 19, wherein the analyzing mechanism is
2 further configured to analyze the library to determine call arguments for the
3 defined symbol name.

1 22. The apparatus of claim 21, wherein analyzing the library to
2 determine call arguments for the defined symbol name is accomplished at runtime
3 by analyzing a current call frame.

1 23. The apparatus of claim 21, further comprising a copying
2 mechanism that is configured to copy call arguments from the proxy sub-routine
3 to a call to the native code sub-routine.

1 24. The apparatus of claim 21, further comprising a returning
2 mechanism that is configured to return a result value from the native code sub-
3 routine to the proxy sub-routine.

1 25. The apparatus of claim 19, wherein operations in the first process
2 are isolated from memory and other system resources belonging to the second
3 process so that an error in the first process does not, one of, corrupt memory
4 belonging to the second process and interfere with the second process in any way .

1 26. The apparatus of claim 19, further comprising an inter-process
2 communication mechanism that is configured so that the proxy sub-routine and
3 the native code sub-routine can communicate.

1 27. The apparatus of claim 19, further comprising an address width
2 translating mechanism that is configured to translate an address from a first
3 address width in the computer program to a second address width in the native
4 code sub-routine.